

Energy and Performance—Can a Wimpy-Node Cluster Challenge a Brawny Server?

Daniel Schall
schall@cs.uni-kl.de

Theo Härder
haerder@cs.uni-kl.de

Databases and Information Systems Group
University of Kaiserslautern, Germany

ABSTRACT

Traditional DBMS servers are usually over-provisioned for most of their daily workloads and, because they do not show good energy proportionality, waste a lot of energy while underutilized. A cluster of small (wimpy) servers, where the number of nodes can dynamically adjust to the current workload, might offer better energy characteristics for these workloads. Yet, clusters suffer from “friction losses” and may not be able to quickly adapt to the workload, whereas a single, brawny server delivers performance instantaneously.

In this paper, we compare a small cluster of lightweight nodes to a single server in terms of performance and energy efficiency. We run several benchmarks, consisting of OLTP and OLAP queries at variable utilization to test the system’s ability to adjust to the workloads. To quantify possible energy saving and its conceivable drawback on query runtime, we evaluate our implementation on a cluster as well as on a single, brawny server and compare the results w.r.t. performance and energy consumption. Our findings confirm that—based on the workload—energy can be saved without sacrificing too much performance.

1. INTRODUCTION

Saving energy is a concern in all areas of IT. Studies have shown that single servers have potential for energy optimizations, but, in general, the best performing configuration is also the most energy efficient one [20]. This observation stems from the fact that the power spectrum between idle and full utilization of a single server is narrow and 50% of its power is already consumed at idle utilization [2].

Today’s server hardware is not energy-proportional; at low utilization, hardware—mainly main memory and storage drives—consumes a significant amount of power. Hence, about half of the maximum power of a server is already going to waste when idle. Automatically scaling systems down when idle, thus preventing high idle power consumption of today’s servers is the main focus of energy proportionality. Unfortunately, current hardware is not energy proportional.

Several components such as CPUs are able to quickly change into sleep states, requiring less energy, when idle. Other components, especially the two main energy consumers of DBMSs, main memory and external storage, exhibit bad energy characteristics.

Therefore, better energy efficiency cannot be achieved with current, centralized solutions. This observation also holds for traditional DBMSs, composed of a single server with huge main memory and lots of storage drives attached. In contrast to centralized, brawny servers, a scale-out cluster of lightweight (wimpy) servers has the ability to shutdown single nodes independently. At an abstract level, this enables the cluster to dynamically add or remove storage and processing power based on the cluster’s utilization.

With cloud computing, elastic systems have emerged that adapt their size to the current workload. While stateless or lightweight systems can easily increase or reduce the number of active computing nodes in a cluster, a database faces much more challenges due to high interactions among the nodes and fast reachability of DB data.

Similar to cloud-based solutions, we hypothesize that a cluster of nodes may adjust the number of active (power-consuming) nodes to the current demand and, thus, approximate energy proportionality.

Based on these observations, we developed WattDB, a research prototype of a distributed DBMS cluster, running on lightweight, Amdahl-balanced nodes using commodity hardware. The cluster is intended to dynamically shrink and expand its size, dependent on the workload. Although the cluster may not be as powerful as a monolithic server, for typical workloads, we expect our system to consume significantly less energy.

Reconfiguring a cluster to dynamically match the workload requires data to be moved from node to node to balance utilization. Yet, copying data is time-consuming and adds overhead to the already loaded cluster. Reducing both, time and overhead, is crucial for an elastic DBMS.

In this paper, we compare a single, brawny server with a cluster of wimpy nodes under OLTP and OLAP workloads, running TPC-H and TPC-C respectively. First, we give an overview of recent research addressing partitioning, elasticity, and energy efficiency of DBMSs in Section 2. In the following section, we introduce important aspects of our energy-proportional database cluster, called *WattDB*. Section 4 contains the results of several empirical experiments and compares energy use and performance of our cluster to those of a brawny server. In Section 5, we summarize the main issues of our work and give some conclusions.

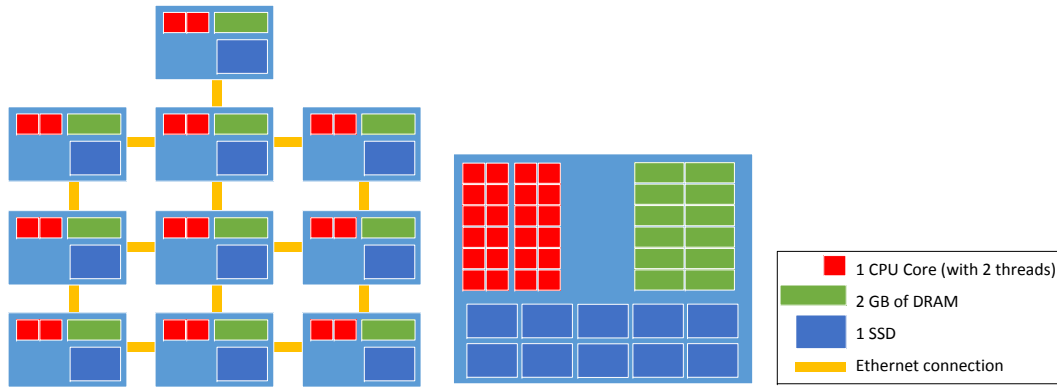


Figure 1: The 10-node cluster compared with the brawny server

2. RELATED WORK

Reducing energy consumption of servers and dynamic re-configuration are all subject to a variety of research approaches. For the reason, we give a short overview of related works in three fields that serve as a building blocks of our research.

2.1 Dynamic Clustering

Traditional clustered DBMSs do not dynamically adjust their size (in terms of the number of active nodes) to their workload. Hence, scale-out to additional nodes is typically supported, whereas the opposite functionality, shrinking the cluster and centralizing the processing—the so-called scale-in—is not. Recently, with the emergence of clouds, a change of thinking occurred and dynamic solutions became a research topic.

In his PhD thesis [8], Sudipto Das implemented an elastic data storage, called Elastras, able to dynamically grow and shrink on a cloud. As common in generic clouds, his work is based on decoupled storage where all I/O involves network communication. *Key Groups*, an application-defined set of records frequently accessed together, can be seen as dynamic partitions that are often formed and dissolved. By distributing the partitions among nodes in the cluster, both performance and cost can be controlled.

A lot more data management systems working on a cloud have been proposed. In [5], Brantner et al. designed a DBMS using Amazon S3 as storage and running on top. Lomet et al. [13] divided the database into two layers, one transactional and one persistence component that can run independently.

In [1], Armbrust et al. propose a scalable storage layer supporting consistency and dynamic scale-out/in called SCADS. Objects in SCADS are stored in logical order. Hot, i.e., frequently accessed objects are distributed among disks to improve access latencies and mitigate bottlenecks. The system was also extended to automatically adjust to workload changes and autonomously redistribute data.

Besides relational approaches, other implementations relax traditional DBMS properties to gain performance and simplify partitioning. Yahoo PNUTS [7], Bigtable [6], and Cassandra¹, are example of systems sacrificing transaction or schema support and query power [1]. Instead of arbitrary

¹<http://cassandra.apache.org/>

access patterns on the data, only primary key accesses to a single record are supported [21].

As an improvement, Amazon’s SimpleDB² allows transactions to access multiple records, but limits accesses to single tables. Moreover, most current scalable data storage systems lack the rich data model of an RDBMS, which burdens application developers with data management tasks. Yet, no *fully-autonomous, clustered DBMS* exists which can provide ACID properties for transactions and SQL-like queries while dynamically adjusting its size to the current workload.

2.2 Energy Optimizations

Lang et al. [12] have shown that a cluster suffers from “friction losses” due to coordination and data shipping overhead and is therefore not as powerful as a comparable heavy-weight server. On the other hand, for moderate workloads, i.e., the majority of real-world database applications, a scale-out cluster can exploit its ability to reduce or increase its size sufficiently fast and, in turn, gain far better energy efficiency.

In [15], we already explored the capabilities and limitations of a clustered storage architecture that dynamically adjusts the number of nodes to varying workloads consisting of simple *read-only page requests* where a large file had to be accessed via an index³. We concluded that it is possible to approximate energy proportionality in the storage layer with a cluster of wimpy nodes. However, attaching or detaching a storage server is rather expensive, because (parts of) datasets may have to be migrated. Therefore, such events (in appropriate workloads) should happen on a scale of minutes or hours, but not seconds.

In [14], we have focused on the query processing layer—again for varying workloads consisting of two types of *read-only SQL queries*—and drawn similar conclusions. In this contribution, we revealed that attaching or detaching a (pure) processing node is rather inexpensive, because repartitioning and movement of data is not needed. Hence, such an event can happen in the range of a few seconds—without disturbing the current workload too much.

We substantially extended the kind of DBMS processing supported by WattDB to *complex OLAP / OLTP work-*

²<http://aws.amazon.com/simpledb/>

³Starting our WattDB development and testing with rather simple workloads facilitated the understanding of the internal system behavior, the debugging process, as well as the identification of performance bottlenecks.

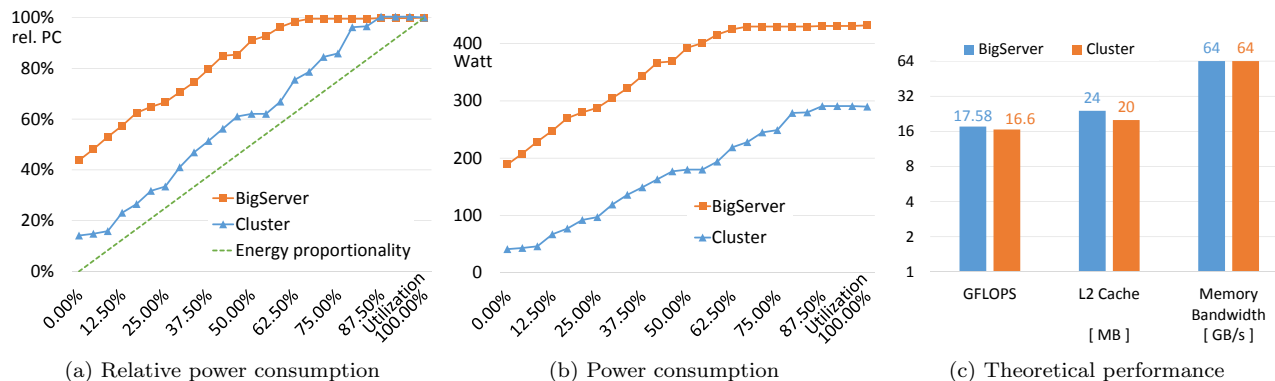


Figure 2: Power consumption and performance figures for both systems

loads consisting of read-write transactions in [16]. For this purpose, we refined and combined both approaches to get one step closer to a fully-featured DBMS, able to process OLTP and OLAP workloads simultaneously. In this work, we were able to trade performance for energy savings and vice versa. Yet, we found out that the adaptation of the cluster and the data distribution to fit the query workload is time-consuming and needs to be optimized.

As discovered before, a single-server-based DBMS is far from being energy-proportional and cannot process realistic workloads in an energy-efficient way. Our previous research indicates that a cluster of lightweight (wimpy) servers, where nodes can be dynamically switched on or off, seems more promising. In this paper, compare our scale-out cluster to a big server to quantify possible energy savings and to discover promising workloads.

3. CLUSTER VS. BIG SERVER

Our cluster hardware consists of n (currently 10) identical nodes, interconnected by a Gigabit-ethernet switch. Each node is equipped with an Intel Atom D510 CPU (with two threads using HyperThreading) running at 1.66 GHz, 2 GB of DRAM and an SSD for data storage. The configuration is considered Amdahl-balanced [19], i.e., balanced w.r.t. I/O and network throughput on one hand and processing power on the other. By choosing commodity hardware with limited data bandwidth, GB-Ethernet wiring is sufficient for interconnecting the nodes. All nodes can communicate directly.

To compare performance and energy savings, we ran the same experiments again on a single, brawny server. This server has two Intel Xeon X5670 processors with 24 GB of RAM and 10 SSDs.⁴ Each CPU has 12 cores and 24 threads (using HyperThreading), running at 2.93 GHz.

Figure 1 sketches the cluster with 10 nodes and the big server. For comparison, we have highlighted the main components (CPU cores, main memory and disk) inside the nodes as well as the communication network.

Each wimpy node consumes $\sim 22 - 26$ Watts when active (based on utilization) and ~ 2.5 Watts in standby. The interconnecting network switch consumes 20 Watts and is included in all measurements.

⁴For a fair comparison with the wimpy nodes, we have reduced the RAM to 24GB, although the server can handle much more. Yet, with more main memory, the power consumption of the server would also be much higher.

In its minimal configuration—with only one node and the switch running and all other nodes in standby—the cluster consumes approx. 65 Watts. This configuration does not include any disk drives, hence, a more realistic minimal configuration requires about 70 Watts. In this state, a single node is serving the entire DBMS functionality (storage, processing, and cluster coordination). With all nodes running at full utilization, the cluster will consume ~ 260 to 280 Watts, depending on the number of disk drives installed.

This is another reason for choosing commodity hardware which uses much less energy compared to server-grade components. For example, main memory consumes ~ 2.5 Watts per DIMM module, whereas ECC memory, used in the brawny server, consumes ~ 10 Watts per DIMM.

The power consumption of the brawny server (with 10 SSDs) ranges from ~ 200 Watts when idle to ~ 430 Watts at full utilization.⁵ In theory, the systems should show similar performance. All nodes in the cluster come with 16.6 (10x1.66) GFLOPS, whereas the performance of the big server is rated with 17.6 GFLOPS. Furthermore, L2 caches and memory bandwidth of both systems are similar and the same number of disks is installed. Figure 2 depicts power consumption and performance figures, as given in the product sheets, for both systems.

3.1 DBMS Software

By the time, research gained interest in energy efficiency of database servers, no state-of-the-art DBMS was able to run on a dynamically adapting cluster. To test our hypotheses (see Section 1), we developed *WattDB* that supports SQL query processing with ACID properties, but is also able to adjust to the workload by scaling out or in, respectively.

To enable a fair comparison, the same software is running on the cluster and the big server. On the latter, the dynamic features of *WattDB* are not needed and are therefore disabled.

The smallest configuration of *WattDB* is a single server, hosting all database functions and acting as endpoint to DB clients. This server is called *master node*. DB objects (tables, partitions) and query evaluation can be offloaded to arbitrary nodes in the cluster to relieve the node, but it will always act as the coordinator and client endpoint.

Some of the key features and design considerations of *WattDB* are explained in the following.

⁵These measurements include only 24 GB of DRAM as previously explained.

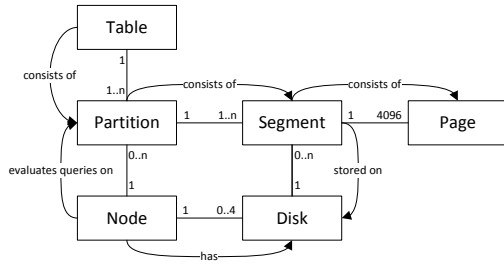


Figure 3: Database schema

3.2 Dynamic Query Processing

To run queries on a cluster of nodes, distributed query plans are generated on the master node. Except data access operators which need local access to the database’s records, all query operators can be placed on remote nodes. Running query operators on a single node does not involve network communication among query operators, because all records are transferred via main memory. Distributing operators implies shipping of records among nodes and, hence, introduces network latencies. Additionally, the bandwidth of the Gigabit Ethernet, which we are using for our experiments, is relatively small, compared to memory bandwidth.

To mitigate the negative effects of distribution, WattDB is using *vectorized volcano-style query operators* [9, 4]; hence, operators ship a set of records on each call. This reduces the number of calls between operators and, thus, network latencies. To further decrease network latencies, buffering operators are used to prefetch records from remote nodes. *Buffering operators* act as proxies between two (regular) operators; they asynchronously prefetch records, thus, hiding the delay of fetching the next set of records.

In WattDB, the query optimizer tries to put pipelining operators⁶ on the same node to minimize latencies. Offloading pipeline operators to a remote node has little effect on workload balancing and, thus, does not pay off. Instead, blocking operators⁷ may be placed on remote nodes to equally distribute query processing. They generally consume more resources (CPU, main memory) and are therefore prime candidates for workload balancing in the cluster.

3.3 Dynamic Reorganization

The master node is coordinating the whole cluster. It is globally optimizing the query plans, whereas regular nodes can locally optimize their part of the plan. Furthermore, it takes nodes on- and offline and decides when and how DB tables are (re)partitioned.

Every node is monitoring its utilization: CPU, memory consumption, network I/O, and disk utilization (storage and IOPS). Additionally, performance-critical data is collected for each database partition, i. e., CPU cycles, buffer page requests and network I/O. With these figures, we can correlate the observed utilization of cluster components to (logical) database entities. Hence, both types of data are necessary to identify sources of cluster imbalance. We use the performance figures of the components to identify their over- or under-utilization. In addition, activity recording of database

⁶Pipelining operators can process one record at a time and emit the result, e. g., projection operators.

⁷Blocking operators need to receive all records, before they can emit the first result record, e. g., sorting operators.

entities is needed to determine the origin of the cluster’s imbalance. For this reason, the nodes send their recording every few seconds to the master node.

The master checks the incoming performance data to predefined thresholds—with both upper and lower bounds. If an overloaded component is detected, it will decide where to distribute data and whether to power on additional nodes and resume their cluster participation. Similar, underutilized nodes trigger a scale-in protocol, i. e., the master will distribute the data (processing) to fewer nodes and shut-down the nodes currently not needed. Decisions, what data to migrate and where, are done based on the current utilization of the nodes, the expected query workload, and the estimated cost, it will take to migrate data between nodes.

In WattDB, we have implemented different policies regarding the scale-out behavior. First, each node in the cluster stores data on local disks to minimize network communication. If storage space of a node is in short supply, database partitions are split up on nodes with free space.

Second, WattDB tries to keep the I/O rate for each storage disk in a certain range. Underutilized disks are eligible for additional data—either newly generated by INSERT operations or migrated from overloaded disks. Utilization among storage disks is first locally balanced on each node, before an allocation of data from/to other nodes is considered.

Third, each node’s CPU utilization should not exceed the upper bound of the specified threshold (80%). As soon as this bound is violated for a node, WattDB first tries to offload query processing to underutilized nodes.⁸ If the overload situation cannot be resolved by redistributing the query load, the current data partitions and their node assignments are reconsidered. When a partition causing the overload is identified, it is split according the partitioning scheme applied, where affected segments are moved to other nodes [18]. For underutilized nodes, an inverse approach is needed. A scale-in protocol is initiated, which quiesces the involved nodes from query processing and shifts their data partitions to nodes currently having sufficient processing capacity.

Similar rules exist for network and memory utilization, e. g., if the working sets of the transactions become too big for the DB buffer, repartitioning is triggered. WattDB makes decisions based on the current workload, the course of utilization in the recent past, and the expected future workloads [11]. Additionally, workload shifts can be user-defined to inform the cluster of an expected change in utilization.

Cost of reorganization Moving data is an expensive task, in terms of energy consumption and performance impact on concurrently running queries. Data reorganization binds some computing resources, which would be needed to optimally process the query workload. This resource contention leads to fewer resources for the workload and, in turn, reduces query throughput. However, the reorganization cost should amortize by reducing the energy consumption of subsequent queries. Though it is difficult to calculate the exact energy consumption of a data move operation with respect to the impact of running queries, the energy cost can be estimated with the duration of the move operation and the (additional) power consumption. Hence, moving 1 GByte of data to a dedicated node with 25 Watts power consumption will require approximately 10 seconds and 250 Joules.

⁸This works well for operators like **SORT**, **GROUP**, and **AGGREGATE**.

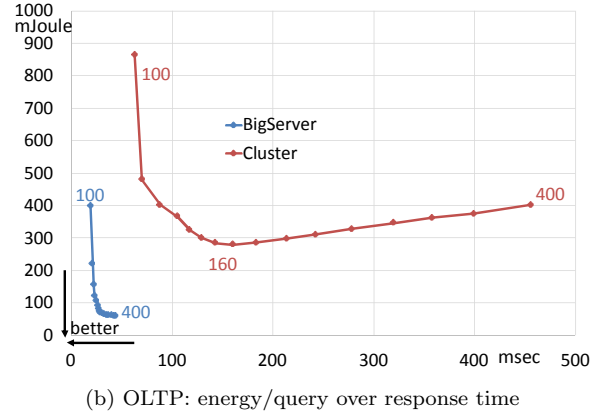
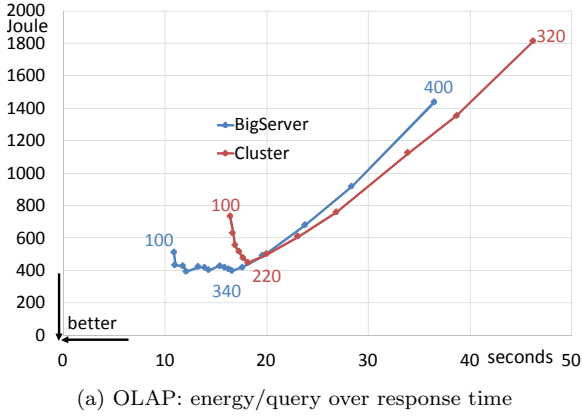


Figure 4: Peak Performance and energy consumption for both systems

In order to save energy, reconfiguration overhead needs to pay off by reducing query runtimes in the future. Likewise, scale-in must trigger when the cluster is able to handle the workload with less nodes. To estimate the impact of reorganization, WattDB relies on a simplified cost model where upcoming workload predictions and maintenance costs are calculated.

3.4 Power Measurement

We have developed a measurement device, capable of monitoring the power and energy consumption of each node in the cluster. The device is also able to these metrics of the big server. This device sends the stream of measurements to a connected PC, running the monitoring software. The monitoring software can further capture the number of active database nodes and the total throughput and response times of queries during the tests. This computer is controlling the benchmark execution by submitting queries to the master node; thus, it enables fine-grained monitoring in correlation with the benchmark runs. The measurement frequency of the device reaches up to 100 Hz; hence, we are able to determine power use in high resolution. A detailed description of the measurement device can be found in [10].

The energy measurement device is only used for external monitoring; WattDB cannot use the measurements to improve its energy efficiency. Internally, the DBMS is working with estimates to determine overall power consumption.

4. EXPERIMENTS

To compare the energy consumption of our cluster to that of a traditional DB server, we have processed OLAP and OLTP workloads on both platforms. We have run performance-centric benchmarks first, to assess peak performance of both systems. Next, we have evaluated energy-centric benchmarks to identify energy-efficiency potential of the big server and the cluster. In the following, we first describe the experimental setup, before we present our results.

4.1 Experimental Setup

For all experiments, using OLTP and OLAP, we have set up the systems as previously described. A separate server, directly connected to the master node and the big server, respectively, is used as the benchmark driver, submitting queries to the cluster as well as monitoring response time and

throughput. The previously introduced power measurement device is also hooked up to the benchmark driver to correlate all measurements with energy consumption.

OLAP workloads: For measuring OLAP performance and energy efficiency, we are using the well-known TPC-H benchmark with a scale factor of 300; hence, 300 GByte of raw data are generated. Due to additional indexes and storage overhead, the final DB has approx. 460 GByte of raw data. On the centralized server, small tables are stored on a single disk, whereas larger ones, e.g., the *LINEITEM* and *ORDERS* tables, are partitioned and distributed among all disks to increase access bandwidth and to parallelize processing on partitions.

On the cluster, the *REGION* and *NATION* tables are replicated to all nodes, while the other tables are partitioned and distributed equally among the nodes. In static benchmarks, no repartitioning occurs, even if the initial distribution leads to hotspots in the data, that impact the node's performance. If dynamic features of WattDB are enabled, the DBMS will automatically repartition as previously described.

OLTP Workloads: For online transaction processing, we are running the TPC-C benchmark on the systems with a scale factor of 1000. Hence, a thousand warehouses were generated on the cluster, consisting of about 100 GB of data. Due to additional indexes and storage overhead, the final DB has approx. 200 GByte of raw data in the beginning of the experiments.

4.2 Performance-centric benchmark

First, to evaluate the *peak performance* of both systems, we run performance-centric benchmarks similar to TPC-C and TPC-H on the cluster and the big server. We repeated the experiments with a varying number of parallel DB clients in order to estimate a saturation point, i.e., how many parallel queries the systems can process—without entering an overload state. In Figure 4a, the OLAP benchmark results are depicted. The x-axis shows the response time in seconds, while the y-axis illustrates the energy consumption per query. The numbers on the individual graphs annotate the number of parallel DB clients for this curve progression.

From the figure, we can conclude that the big server handles queries generally faster than the cluster, it also exhibits better energy efficiency. When raising the number of clients, the brawny machine takes longer to respond to queries, yet

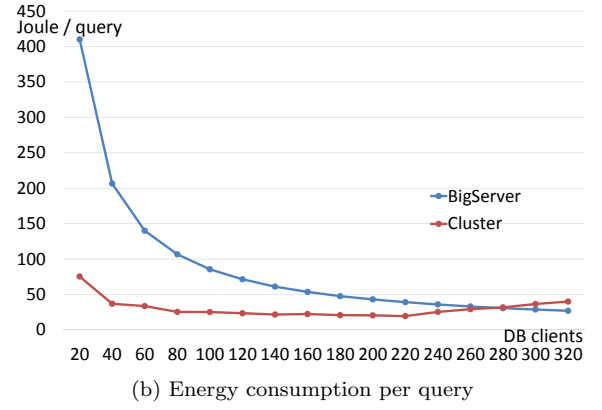
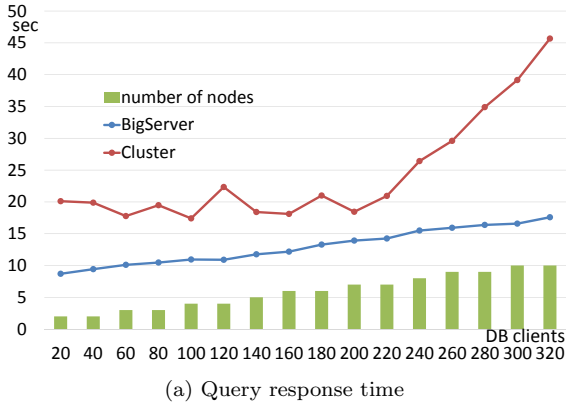


Figure 5: Performance and energy consumption for varying OLAP utilizations

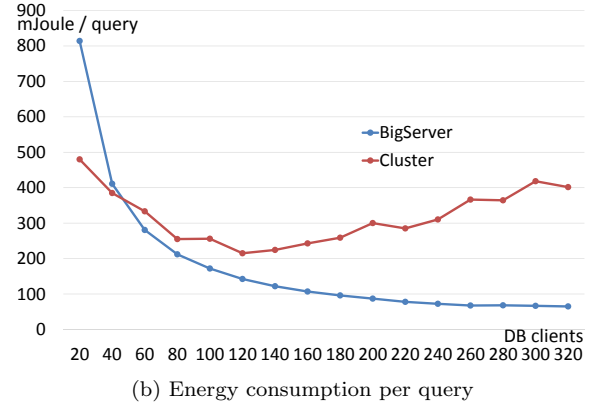
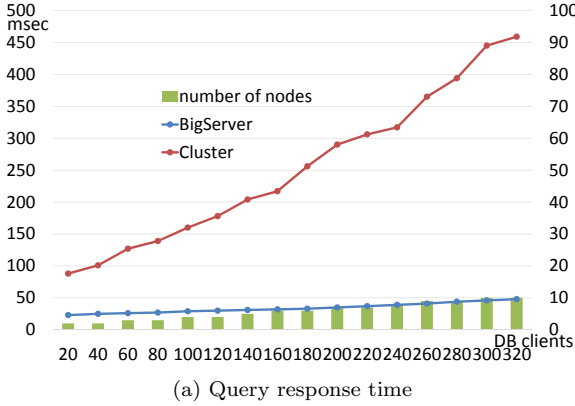


Figure 6: Performance and energy consumption for varying OLTP utilizations

up to 340 clients, runtimes only slightly increase. After that point, the server seems saturated and runtimes start to build up. Consequently, energy consumption per query rises.

The cluster handles medium-sized workloads (up to 220 clients) slightly worse than the cluster. Yet, more than 220 clients seem to overload the cluster as runtimes and energy consumption quickly increase. When stressing the cluster with more than 340 clients, the database crashes due to shortage of main memory.

Figure 4b illustrates the results of the same experiments repeated using OLTP queries. The results reveal that the big server is much better suited for OLTP than the cluster, as it exhibits lower query response times and also less energy consumption. Query response times on the brawny server increase only slightly with the number of DB clients and the system does not show saturation at all. Consequently, energy consumption per query improves continuously.

The cluster is saturated with 160 clients; when further increasing the number of parallel queries, response times start to increase faster.

Analyzing access patterns of both, OLTP and OLAP, the different performance figures are explainable: OLAP queries read huge amounts of records, join them with (small) fact tables, and then group and aggregate the results to satisfy analytical inquiries. Hence, the reading part of these queries can run in parallel on all partitions, speeding up the query linearly with the number of disks, CPUs, and/or nodes. After having fetched the qualified records, the joins with the

fact tables can also run concurrently. The final grouping and aggregation steps can be pre-processed locally for each of the parallel streams and quickly aggregated into a final result. Hence, this kind of access pattern seems to well fit both, a single, multi-core machine with lots of disks and memory but also a cluster of independent nodes, exchanging query results via network. In [18], we have analyzed the abilities of a cluster to process that kind of workloads in greater detail.

In contrast, OLTP queries touch very little data, but update records frequently. Since writers need to synchronize to avoid inconsistencies, lock information must be shared among all nodes involved. A centralized machine keeping the lock table in main memory is able to synchronize transactions much quicker than a cluster, needing to exchange lock tables among nodes. Further, OLTP query operators modifying records cannot be offloaded to other nodes. Therefore, the query plan for transactional workloads is much more rigid than OLAP queries.

In summary, it is comprehensible, that a cluster of nodes is better suited for OLAP workloads than OLTP.

4.3 Energy-centric benchmark

After evaluating the peak performance of both configurations, we ran experiments representing average, real-world workloads. Because DB servers are typically heavily underutilized, as mentioned earlier, we modified the benchmark driver to submit queries *at timed intervals*.

Workload scaling: In each experiment, we have spawned a number of OLTP or OLAP clients, sending queries to the database. Each client sends a query in a specified interval. If the query is answered within the interval, the next query is not initiated immediately, but at the start of the subsequent interval. If the query is not finished within the interval, the client waits for the answer until sending the next query. In this way, each DB client generates its share of utilization. The database has to answer queries quickly enough to satisfy the DB clients, but there is no reward for even faster query evaluation. It is important to delay query submission of the clients, because we are not interested in maximizing throughput, but instead, want to adjust the DBMS to a given workload, using an optimal number of nodes.⁹

Before each workload changes, the cluster is manually re-configured to best match the expected workload. We let the benchmarks run for a short warm-up time prior to measuring energy efficiency and performance to eliminate start-up cost and to identify maximum energy savings potential.

OLAP: In this experiment, we vary the number of parallel clients between 20 and 320. As before, we are using TPC-H queries on a SF300 dataset. To control utilization, the clients send queries at an interval of at least 20 seconds. Whatever comes last, the query result or the end of the interval, is the trigger for the next query.

Figure 5 illustrates the results for the energy-centric OLAP benchmark. The left side depicts query response times of the brawny server and the wimpy cluster. As expected, the centralized machine handles queries faster than the cluster, even faster than the target response time of 20 seconds per query. Therefore, the server is idle for longer time periods, still consuming energy.

The cluster is meeting the target response times quite well, except for higher utilization, as observed earlier. After about 220 parallel clients, query performance starts to drop and runtimes build up.

Comparing energy consumption per query of both systems, the cluster delivers far better results for *average utilizations*. Due to the cluster’s scale-out and adaptation to the necessary number of nodes, its energy consumption per query stays at the same level almost the entire time, regardless of utilization. Only at high workloads, energy consumption increases because of lengthy query runtimes.

The big server, with more or less static power consumption over the whole utilization spectrum, delivers bad energy efficiency for low and moderate workloads. Only at high utilization, when all the processing power of the server is needed, its energy consumption per query pushes below that of the cluster.

From this experiment, we conclude that the cluster seems to be better fit for moderate OLAP workloads than the big server.

OLTP: We repeated the same experiment using OLTP queries from the TPC-C benchmark. Prior, the corresponding dataset was generated with a scale factor of 1,000. Identical to the OLAP benchmark, we scaled the DB clients between 20 and 320. Each client was waiting 40 ms between queries to simulate low and moderate workloads too.

Figure 6 plots the results of the energy-centric OLTP benchmark run. Whereas the big server exhibits query re-

sponse times between 30 and 50 milliseconds, the cluster performs with processing times between 50 and 450 ms much worse. Apparently, the cluster is not well suited to process update-intensive OLTP workloads.

On the other hand, energy efficiency of the cluster is much better at low workloads. While the big server consumes between 200 and 800 mJoule per query, the cluster only needs about 150 mJoule/query.

In conclusion, we have constituted a tradeoff between performance and energy consumption on the cluster. By reducing the number of nodes, both power consumption and peak performance are lowered. For moderate workloads, lower performance is tolerable and, thus, energy efficiency can be improved.

4.4 Dynamic workloads

As previously described, the limiting factor for dynamic repartitioning is the migration cost, i.e., the performance impact and time it takes to move data between nodes. To estimate its impact on the cluster’s elasticity, we have run experiments on a dynamically adapting cluster. Similar to the previous tests, we are running a mix of workloads against the cluster, ranging from low utilization up to heavy workloads. In this experiment, the cluster is given no warm-up times to adjust itself to a given task; instead, we are monitoring performance and energy consumption continuously.

Workloads change every 5 minutes, starting with a moderate workload of 20 database clients, sending OLTP or OLAP queries respectively. The workload pattern is depicted underneath all result figures.

To quantify the importance of forecastable workloads, we have run the same workloads on the cluster twice—and for comparison once on the big server. The first run on the cluster hits the database unexpectedly; WattDB will have to reactively adjust to the workload. After that, the same benchmark is used again, this time informing the database of upcoming workloads (30 minutes in advance). Hence, the database may use the information to proactively adjust. As stated by Kramer et. al. [11], database workloads are often repetitive and, therefore, quite easy to forecast.

In the following, results of the benchmark runs are discussed separately for OLAP and OLTP. Results for the big server are depicted on the left side of figures 7 and 8. In the middle part, the plots represent benchmark results measured with the non-forecasting cluster. The right side illustrates results of the same experiment using forecasting. The top-most plot in every column draws the average query response time. The target response time of 20 seconds is included to expose the load-dependent response time deviations in the various experiments. To characterize the varying size of the cluster, the number of active nodes is visualized. Underneath, the course of the overall power consumption is shown for all three experiments. The resulting average energy consumption per query is plotted in the graphs below—to contrast it to the power consumption. The last charts in each column visualizes the workload mix (which was the same for all three experiments).

OLAP (Big Server): Figure 7, leftmost column, shows the results for TPC-H queries on the big server. The brawny server does not exhibit transition times between workloads, since no reconfiguration is needed on this single-node system. Query runtimes are fast, always beating the target response time. Yet power consumption is constantly high,

⁹Otherwise, the whole benchmark would degenerate to a simple performance-centric evaluation, which is not what we intended.

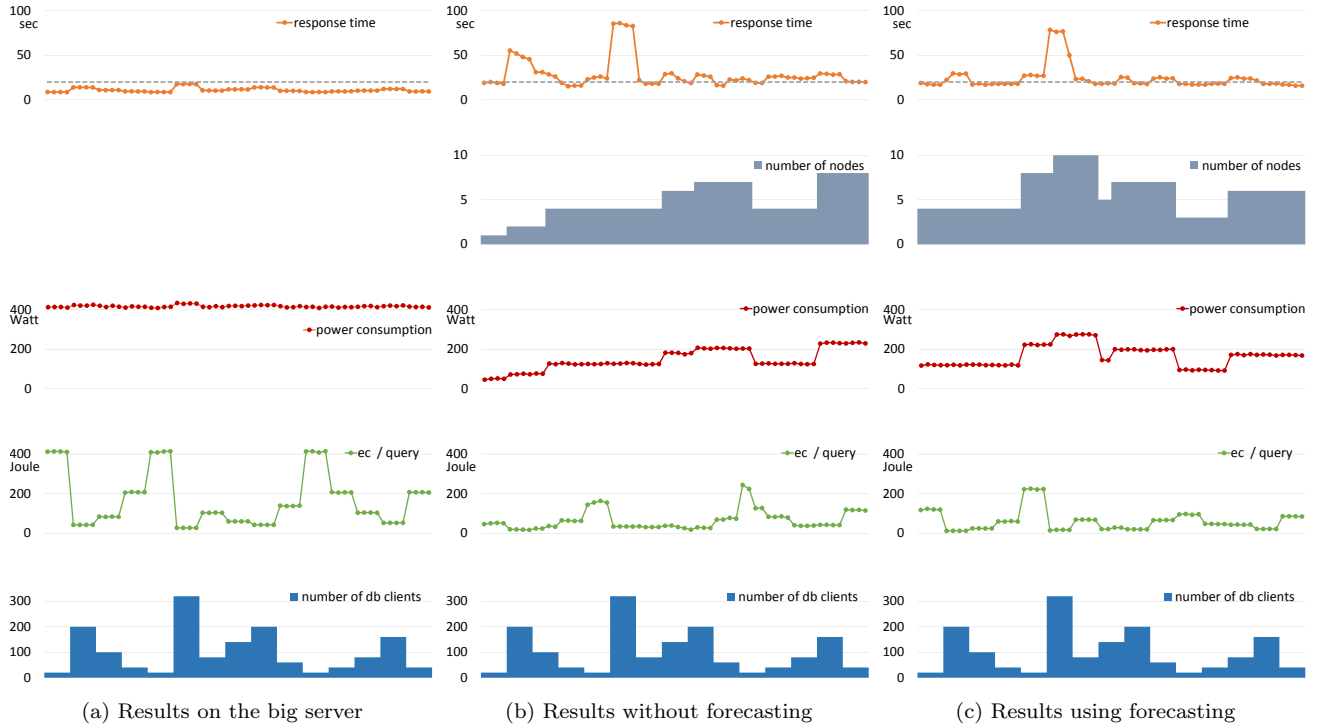


Figure 7: Dynamic OLAP workload on the big server and the cluster

regardless of utilization, as already observed in earlier experiments. Average energy consumption per query is comparably high, although query runtimes are low. Because this benchmark is energy-centric, faster query runtimes do not lead to better results.

OLAP (non-forecasting): In the middle column of Figure 7, TPC-H results for the cluster, not using forecasting, are depicted. The number of nodes in the cluster jitters heavily, as the system tries to adjust itself to the current workload. Reconfiguration takes time, e.g., migrating from 2 to 4 nodes requires each of the two source nodes to ship about 100 GB of data to one of the targets, hence, it takes about 20 minutes to repartition. Therefore, query response times in this benchmark experiment are highly fluctuating and often missing the predefined deadline. Yet, as we have shown in the previous experiments, the cluster, in theory, should be able to handle most of the workloads within the deadline. Due to the high additional reconfiguration overhead, the nodes are overloaded. Therefore, query runtimes and average energy consumption per query remain high.

OLAP (forecasting): In this benchmark, we inform the cluster of workload changes present in the next 30 minutes. Hence, instead of only reacting to workload changes, WattDB can now prepare for upcoming load. The plots on the rightmost column of Figure 7 illustrate the results. In comparison to the first run on the cluster, response times are generally lower and more often passing the deadline. Because the cluster prepares for heavy workloads in advance by scaling out to more nodes, the number of nodes is also larger in average, resulting in increased power consumption. Resulting energy consumption per query shows a mixed picture. For low utilizations, but more nodes running to prepare for upcoming events, energy consumption

is higher compared to the non-forecasting version. On the other hand, for higher utilizations, thanks to in-advance preparations, query runtimes are lower and exhibit overall better energy efficiency.

When comparing the big server with the cluster, we can conclude that the server is more powerful and exhibits lower query response times. On the other hand, the cluster is more energy efficient, especially during low and moderate utilization, due to its adaptation to the workload. The cluster benefits from scale-in, when performance is not needed. This translates to a steadily varying power consumption (according to the cluster size), whereas the server displays a more or less constant one. For OLAP workloads, the cluster seems like an eligible alternative to a big server.

OLTP (Big Server): After running OLAP benchmarks, we have repeated the same dynamic workload with OLTP clients on the TPC-C dataset. Figure 8 illustrates our results for energy-centric OLTP benchmark experiments. The left column depicts those for the big server.

OLTP (non-forecasting): The middle column of Figure 8 summarizes our results for the benchmark run on a non-forecasting cluster. Obviously, the response times shown are high. Because the cluster is forced to permanently repartition, response times and, in turn, energy efficiency are further worsened. Because the cluster can only react to workload changes, rebalancing starts after big workloads hit the cluster. As discussed for the OLAP benchmark, this puts too much stress on the nodes and notably slows down query processing. Compared to the big server, query response times are much higher for high utilizations. Yet, power and energy consumption are lower. Therefore, the cluster delivers better energy efficiency overall—if longer query response times are deemed acceptable.

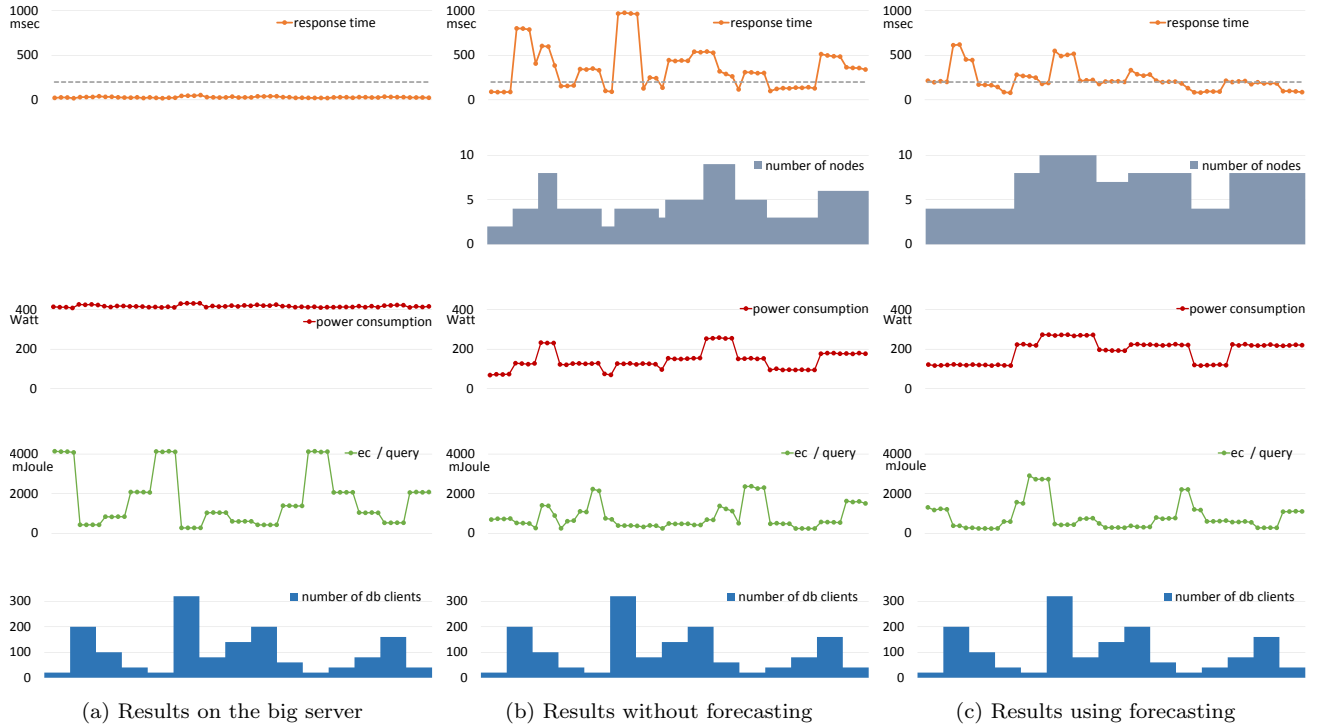


Figure 8: Dynamic OLTP workload on the big server and the cluster

OLTP (forecasting): The right-most column of Figure 8 plots the OLTP benchmark results on the cluster using forecasting. Compared to the previous benchmark, the average number of nodes is higher, because WattDB is preparing for workloads in advance. As a result, query runtimes are more stable and more often pass the deadline. However, power consumption is often higher. Again, overall energy efficiency is characterized by a mixed picture: Due to preparations, lower workloads have worse energy efficiency, but more intense workloads benefit from forecasting by achieving lower energy consumption per query.

Summary Reviewing the results from all benchmarks, we want to extract some condensed numbers to facilitate high-level comparison and to gain a few key observations. For this reason, we have separately computed indicative numbers for the dynamic OLAP and OLTP experiments: total energy consumed (in Watt hours), overall query throughput in units of 10^3 resp. 10^6 , and average energy consumption in Joule resp. mJoule per query. These condensed numbers are visualized in Figure 9, where the logarithmic y-axis should be regarded.

First, the cluster is no match for the big server considering pure performance. The centralized system does not require network communication or synchronization among nodes. Therefore, it can deliver much better query throughput than the cluster, where queries have to be distributed, results have to be collected, and the overall execution of concurrent queries on multiple nodes needs some form of synchronization to ensure ACID properties. All these factors lead to friction losses which slows down query processing.

Second, the cluster handles low and moderate workloads quite well, although the big server is still faster. Yet, the cluster requires less than half of the server’s power (left-

most bars in the figures). Therefore, the cluster needs less energy per query and is more energy efficient, as depicted by the right-most bars in the figures.

Third, dynamic workloads with varying utilization require preparation to adjust the number of nodes to the needs. If workloads are predictable, the cluster exhibits better energy efficiency than the single server while delivering comparable performance. Although, energy consumption of a forecasting cluster is higher, its query performance outweighs the additional wattage.

5. CONCLUSION

In this paper, we have examined the power-saving potential of a clustered DBMS compared to a traditional DBMS based on a single server. An important goal of this paper was to compare performance and energy efficiency of our WattDB cluster to those of a big server. Of course, if peak DBMS performance is required during almost the entire operating time, a single-server approach has no alternative as our performance-centric benchmarks clearly reveal. However, as stated in various studies [3, 17], average utilization figures are far from continuous peak loads. A large share of database or data-intensive applications runs less than an hour close to peak utilization on workdays and is resilient w.r.t. somewhat slower response times. During the remaining time, their activity level is typically in the range of 20–50% and often lower. Therefore, from low- to mid-range workloads, a dynamically adjusting cluster of nodes will consume significantly less power without sacrificing too much performance. Hence, their response time / throughput requirements could be conveniently satisfied by the performance characteristics of our cluster with much less energy use, as confirmed by Figure 7. Hence, the application range,

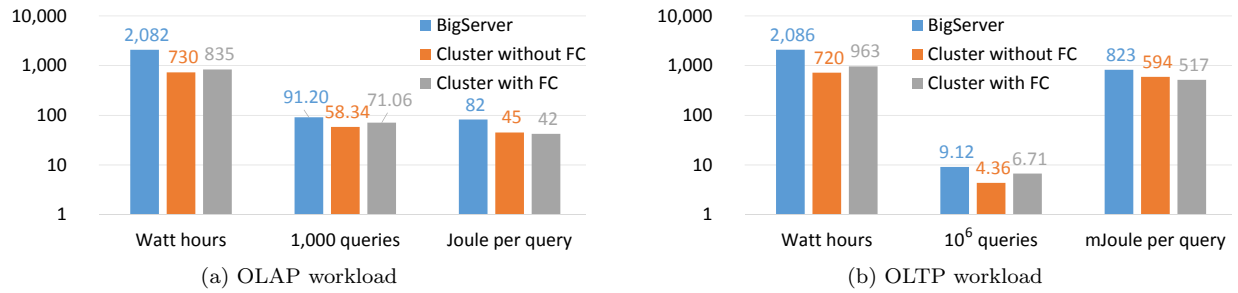


Figure 9: Overall energy consumption, throughput, and average energy consumption per query

where the cluster’s energy efficiency largely dominates that of a single server, has quite some practical benefit.

Especially OLAP workloads, where lots of records need to be read and aggregated without much coordination effort, a cluster seems to be a viable alternative to a single server. On the other hand, when processing OLTP workloads, where transactions need to synchronize continuously, a cluster suffers from high “friction losses” and is a magnitude slower than the centralized approach.

As shown, predictability of workloads and data elasticity are crucial for our approach. Fortunately, typical usage patterns are predictable and a cluster can therefore prepare for upcoming workloads. Thus, dynamically adjusting a cluster to the workload—although time-consuming—is possible.

6. REFERENCES

- [1] M. Armbrust, A. Fox, D. A. Patterson, N. Lanham, B. Trushkowsky, J. Trutna, and H. Oh. SCADS: Scale-Independent Storage for Social Computing Applications. *CoRR*, abs/0909.1775, 2009.
- [2] L. A. Barroso and U. Hölzle. The Case for Energy-Proportional Computing. *IEEE Computer*, 40(12):33–37, 2007.
- [3] L. A. Barroso and U. Hölzle. *The Datacenter as a Computer: An Introduction to the Design of Warehouse-Scale Machines*. Morgan & Claypool Publishers, 2009.
- [4] P. A. Boncz, M. Zukowski, and N. Nes. MonetDB/X100: Hyper-Pipelining Query Execution. In *CIDR*, pages 225–237, 2005.
- [5] M. Brantner, D. Florescu, D. Graf, D. Kossmann, and T. Kraska. Building a Database on S3. In *Proceedings of the 2008 ACM SIGMOD International Conference on Management of Data*, SIGMOD ’08, pages 251–264, New York, NY, USA, 2008.
- [6] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber. Bigtable: A Distributed Storage System for Structured Data. In *Proceedings of the 7th USENIX Symposium on Operating Systems Design and Implementation - Volume 7*, OSDI ’06, pages 15–15, Berkeley, CA, USA, 2006. USENIX Association.
- [7] B. F. Cooper, R. Ramakrishnan, U. Srivastava, A. Silberstein, P. Bohannon, H.-A. Jacobsen, N. Puz, D. Weaver, and R. Yerneni. PNUTS: Yahoo!’s Hosted Data Serving Platform. *Proc. VLDB Endow.*, 1(2):1277–1288, Aug. 2008.
- [8] S. Das. *Scalable and Elastic Transactional Data Stores for Cloud Computing Platforms*. PhD thesis, University of California at Santa Barbara, Santa Barbara, CA, USA, 2011. AAI3495671.
- [9] G. Graefe. Volcano—An Extensible and Parallel Query Evaluation System. *IEEE Trans. on Knowl. and Data Eng.*, 6(1):120–135, Feb. 1994.
- [10] V. Hudlet and D. Schall. Measuring Energy Consumption of a Database Cluster. In *BTW, LNI 180*, pages 734–737, 2011.
- [11] C. Kramer, V. Höfner, and T. Härder. Load Forecasting for Energy-Efficient Distributed DBMSs (in German). *Proc. 42. GI-Jahrestagung 2012, LNI 208*, pages 397–411, 2012.
- [12] W. Lang, S. Harizopoulos, J. M. Patel, M. A. Shah, and D. Tsirogiannis. Towards Energy-Efficient Database Cluster Design. *PVLDB*, 5(11):1684–1695, 2012.
- [13] D. B. Lomet, A. Fekete, G. Weikum, and M. J. Zwillig. Unbundling Transaction Services in the Cloud. *The Computing Research Repository*, abs/0909.1768, 2009.
- [14] D. Schall and T. Härder. Energy-Proportional Query Execution Using a Cluster of Wimpy Nodes. In *SIGMOD Workshops, DaMoN*, pages 1:1–1:6, 2013.
- [15] D. Schall and T. Härder. Towards an Energy-Proportional Storage System using a Cluster of Wimpy Nodes. In *BTW, LNI 214*, pages 311–325, 2013.
- [16] D. Schall and T. Härder. Approximating an Energy-Proportional DBMS by a Dynamic Cluster of Nodes. In *DASFAA Conf., LNCS 8421*, pages 297–311, 2014.
- [17] D. Schall, V. Höfner, and M. Kern. Towards an Enhanced Benchmark Advocating Energy-Efficient Systems. In *TPCTC, LNCS 7144*, pages 31–45, 2012.
- [18] D. Schall and T. Härder. Dynamic Physiological Partitioning on a Shared-nothing Database Cluster. *CoRR*, abs/1407.0120, 2014.
- [19] A. S. Szalay, G. C. Bell, H. H. Huang, A. Terzis, and A. White. Low-Power Amdahl-Balanced Blades for Data-Intensive Computing. *SIGOPS Oper. Syst. Rev.*, 44(1):71–75, 2010.
- [20] D. Tsirogiannis, S. Harizopoulos, and M. A. Shah. Analyzing the Energy Efficiency of a Database Server. In *SIGMOD Conference*, pages 231–242, 2010.
- [21] H. T. Vo, C. Chen, and B. C. Ooi. Towards Elastic Transactional Cloud Storage with Range Query Support. *Proc. VLDB*, 3(1-2):506–514, Sept. 2010.